UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/613,032	07/10/2000	Brian A. LaMacchia	MS#154746.1/40062.65US04	9588

23552 7590 06/30/2005

MERCHANT & GOULD PC
P.O. BOX 2903
MINNEAPOLIS, MN 55402-0903

EXAMINER	
SHIN, KYUNG H	

ART UNIT	PAPER NUMBER
2143	

DATE MAILED: 06/30/2005

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	09/613,032	LAMACCHIA ET AL.	
	Examiner	Art Unit	
	Kyung H. Shin	2143	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *05 April 2005*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-46* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-46* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *10 July 2000* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

1. This action is responding to application papers filed 4/5/2005.

2. Claims **1 - 46** are pending. Independent claims are **1, 13, 20, 26, 28, 40**.

Claim Rejections - 35 USC § 103

4. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

5. Claims **1 - 12, 20 - 25, 28 - 36** are rejected under 35 U.S.C. 103(a) as being unpatentable over **Scheifler et al.** (US Patent No. 6,138,238) in view of **Shrader et al.** (US Patent No. 6,526,513) and further in view of **Berry et al.** (US Patent No. 6,735,758).

Regarding Claims 1, 28, Scheifler discloses a method and a computer program product encoding a computer program of claims determining whether a requested permission, requested by a called code frame, is satisfied within a runtime call stack so

as to allow the called code frame to perform a protected operation, the method comprising: (see Abstract)

a) associating a permission grant object with a first code assembly in the runtime call stack; (see Scheifler col. 4, line 63 - col. 5, line 3)

c) demanding via the permission request object the requested permission from the permission grant object to allow the called code frame to perform the protected operation; (see Scheifler col. 14, lines 41-46)

d) determining whether the requested permission is provided in association with the first code assembly by the permission grant object, responsive to the demanding operation; (see Scheifler col. 11, lines 54-57) and

e) permitting execution of the called code frame to perform the protected operation, if the requested permission is provided in association with the first code assembly. (see Scheifler col. 9, lines 28-37)

Scheifler discloses an access control apparatus utilizing a security policy file, a call code stack, and an executor. (see Scheifler col. 4, lines 51-56: *"Systems and methods...determining access control...based on the source of the code and the principal on whose behalf the code is being executed...regulating code access based on either or both of these factors, the security in computer systems can be enhanced."*)

Scheifler does not disclose the dynamic request/grant of a permission set. However, Shrader and Berry discloses a method, computer program product of

determining whether a requested permission, wherein the permission is at least one of a set of permissions, requested by a called code frame, is satisfied within a runtime call stack so as to allow the called code frame to perform a protected operation, the method comprising:

b) dynamically overriding the set of permissions that is assigned to a permission grant object associated with at least one other code assembly preceding the first code assembly; (see Shrader col. 2, lines 12-17; col. 3, lines 49-53; col. 5, lines 29-37: dynamic permission set, no more checks (i.e. stack walking) made, permission checks terminated; see Berry col. 20, line 64 - col. 21, line 4: parameters for termination of a stack walk, stack walk terminated)

It would have been obvious to one of ordinary skill in the art at the time the invention was made to modify Scheifler to enable the capability of the dynamic permission request/grant as taught by Shrader, and to setup parameters for the termination of a stack walk as taught by Berry. One of ordinary skill in the art would be motivated to modify Scheifler in order to enhance the execution of computer code based on permission sets (see Shrader col. 2, lines 7-17: *"...architecture for extending the Java security model to allow a user ... grant permissions dynamically ... If the user grants the permission, the present invention grants the permission for the ProtectionDomain ... "*), and to employ Berry in order to optimize and accurately maintain profiling information. (see Berry col. 3, lines 36-39: *" ... provide a system in which accurate profiling information*

could be maintained globally for a plurality of processors rather than at the processor level ... ")

Regarding Claims 2, 21, 29, Scheifler discloses the method, the runtime system, and the computer program product of claims wherein the called code frame is included within the first code assembly. (see Scheifler col. 19, line 66 - col. 20, line 2)

Regarding Claims 3, 22, 30, Scheifler discloses the method, the runtime system, and the computer program product of claims wherein the called code frame is included within a lower level code assembly following the first code assembly in the runtime call stack. (see Scheifler col. 20, lines 3-7)

Regarding Claims 4, 31, Scheifler discloses the method and the computer program product of claims comprising: associating a second permission grant object with a second code assembly loaded in the runtime call stack, the second code assembly preceding the first code assembly in the runtime call stack. (see Scheifler col. 18, lines 46-56)

Regarding Claims 5, 32, Scheifler discloses the method and the computer program product of claims, further comprising: determining, whether the requested permission is provided in association with the second code assembly by the second permission grant object. (see Scheifler col. 18, lines 57-65)

Regarding Claims 6, 8, 33, 35, Scheifler does not disclose the dynamic request/grant of a permission set. However, Shrader discloses the method of claim 1 wherein the operation of dynamically overriding comprises:

a) asserting within the first code assembly that a permission grant object associated with the at least one other code assembly preceding the first code assembly need not be evaluated to determine whether a specified permission is satisfied in association with the other code assembly in the runtime call stack, regardless of whether the specified permission is provided by the permission grant object associated with the other code assembly; (see Shrader col. 2, lines 12-17; col. 3, lines 49-53)and

b) permitting execution of the called code frame to perform the protected operation, if the requested permission is a subset of the specified permission. (see Shrader col. 2, lines 12-17; col. 3, lines 49-53)

Regarding Claims 7, 34, Scheifler does not disclose the dynamic request/grant of a permission set. However, Shrader and Berry discloses the method of claim 1 wherein the operation of dynamically overriding comprises:

a) asserting within the first code assembly that a permission grant object associated with the at least one other code assembly preceding the first code assembly does not satisfy a specified permission within the runtime call stack, regardless of whether the specified permission is provided by the permission grant object associated with the other code assembly; (see Shrader col. 2, lines 12-17; col. 3, lines 49-53; col. 5, lines 29-37: dynamic permission set, no more checks (i.e.

stack walking) made, permission checks terminated; see Berry col. 20, line 64 - col. 21, line 4: parameters for termination of a stack walk, stack walk terminated) and

b) preventing execution of the called code frame to perform the protected operation, if the requested permission is a subset of the specified permission. (see Shrader col. 2, lines 12-17; col. 3, lines 49-53; col. 5, lines 29-37: dynamic permission set, no more checks (i.e. stack walking) made, permission checks terminated; see Berry col. 20, line 64 - col. 21, line 4: parameters for termination of a stack walk, stack walk terminated)

It would have been obvious to one of ordinary skill in the art at the time the invention was made to modify Scheifler to enable the capability of the dynamic permission request/grant as taught by Shrader, and to setup parameters for the termination of a stack walk as taught by Berry. One of ordinary skill in the art would be motivated to modify Scheifler in order to enhance the execution of computer code based on permission sets (see Shrader col. 2, lines 7-17), and to employ Berry in order to optimize and accurately maintain profiling information. (see Berry col. 3, lines 36-39)

Regarding Claims 9, 36, Scheifler discloses the method and the computer process of claims wherein the permission object encoded in the code assembly, and the corresponding permission objects encoded in the permission grant object satisfy a common permission interface. (see Scheifler col. 20, lines 3-12)

Regarding Claims 10 - 12, 23 - 25, Scheifler discloses the method, the runtime system, and the computer program product of claims wherein the operation of associating a first permission grant object with a first code assembly comprises: associating the first permission grant object with an individual method, class and module of the first code assembly. (see Scheifler col. 11, line 66 - col. 12, line 5)

Regarding Claim 20, Scheifler discloses a runtime system for determining whether a requested permission, requested by a called code frame, is satisfied within a runtime call stack so as to allow the called code frame to perform a protected operation, the runtime system comprising:

a) a first code assembly loaded into the runtime call stack; (see Scheifler col. 14, lines 5-12)

b) a first permission grant object associated with the first code assembly comprising one or more permissions available to the first code assembly; (see Scheifler col. 14, lines 31-37) and

c) a first permission request object created by the called code frame requesting the requested permission from the first permission grant object, wherein the called code frame is permitted to execute the protected operation if the first permission request object determines from the permission grant object that the requested permission is satisfied by the first code assembly. (see Scheifler col. 15, lines 25-32)

Scheifler does not disclose the dynamic request/grant of a permission set. However, Shrader and Berry discloses a runtime system for determining whether a requested permission of a set of permissions, requested by a called code frame, is satisfied within a runtime call stack so as to allow the called code frame to perform a protected operation, the runtime system comprising:

d) wherein the permission set is dynamically overridden to modify one permission within the set of permissions. (see Shrader col. 2, lines 12-17; col. 3, lines 49-53; col. 5, lines 29-37: dynamic permission set, no more checks (i.e. stack walking) made, permission checks terminated; see Berry col. 20, line 64 - col. 21, line 4: parameters for termination of a stack walk, stack walk terminated)

It would have been obvious to one of ordinary skill in the art at the time the invention was made to modify Scheifler to enable the capability of the dynamic permission request/grant as taught by Shrader, and to setup parameters for the termination of a stack walk as taught by Berry. One of ordinary skill in the art would be motivated to modify Scheifler in order to enhance the execution of computer code based on permission sets (see Shrader col. 2, lines 7-17), and to employ Berry in order to optimize and accurately maintain profiling information. (see Berry col. 3, lines 36-39)

6. Claims **13 - 19, 26, 27, 37- 46** are rejected under 35 U.S.C. 103(a) as being unpatentable over **Scheifler** in view of **Jerger et al.** (US Patent No. 6,345,361).

Regarding Claims 13, 40, Scheifler discloses a method determining whether a requested permission, requested by a called code frame, is satisfied within a runtime call stack so as to allow the called code frame to perform a protected operation, the method comprising:

a) associating a first permission grant object with a first code assembly in the runtime call stack; (see Scheifler col. 14, lines 6-12)

b) associating a second permission grant object with a second code assembly in the runtime call stack; (see Scheifler col. 14, lines 6-12)

e) demanding the requested permission; (see Scheifler col. 14, lines 40-45)

Scheifler does not disclose the generation of an intersection (i.e. common subset) of permissions for a method's code processing software and cache (i.e. save) of the permission set information for later usage without re-generation of permission set information. However, Jerger discloses a method determining whether a requested permission, requested by a called code frame, is satisfied within a runtime call stack so as to allow the called code frame to perform a protected operation, the method comprising:

c) computing a first intersection of permissions provided by the first permission grant object and the second permission grant object; (see Jerger col. 35, lines 40-43; col. 58, lines 37-42)

d) recording the first intersection of permissions to provide a cached permission intersection; (see Jerger col. 35, lines 40-43; col. 58, lines 37-42)and

f) permitting execution of the called code frame if the requested permission is a subset of the cached permission intersection. (see Jerger col. 35, lines 40-43; col. 58, lines 37-42)

It would have been obvious to one of ordinary skill in the art at the time the invention was made to modify Scheifler to process permission sets by the generation and save of an intersection of permissions for a software code method as taught by Jerger. One of ordinary skill in the art would be motivated to modify Scheifler in order to enhance security in the execution of software using permission sets. (see Jerger col. 2, line 65 - col. 3, line 5: *"...automatically compare many different types of permissions...provide sets of predetermined security settings that represent varying levels of trust...associated with a zone... provides a way for the user to configure the permission sets down to a very "fine-grained" level.."*)

Regarding Claims 14, 41, Scheifler discloses the method and a computer program product encoding a computer program of claims further comprising:

a) associating a third permission grant object with a third code assembly in the runtime call stack; (see Scheifler col. 14, lines 31-37)

Scheifler does not disclose the generation of an intersection (i.e. common subset) of permissions for a method's code processing software and cache (i.e. save) of the permission set information for later usage without re-generation of

permission set information. However, Jerger discloses the method of claim 13 further comprising:

b) computing a second intersection of permissions provided by the first permission grant object, the second permission grant object, and the third permission grant object; (see Jerger col. 35, lines 40-43; col. 58, lines 37-42)and

c) recording the second intersection of permissions to provide the cached permission intersection. (see Jerger col. 35, lines 40-43; col. 58, lines 37-42)

It would have been obvious to one of ordinary skill in the art at the time the invention was made to modify Scheifler to compute a second intersection of permissions from a first, second, and third permission grant object as taught by Jerger. One of ordinary skill in the art would be motivated to modify Scheifler in order to enhance security in the execution of software using permissions sets. (see Jerger col. 2, line 65 - col. 3, line 8).

Regarding Claim 15, 42, Scheifler discloses the method of claim 1 wherein the called code frame is included within the first code assembly. (see Scheifler col. 19, line 66 - col. 20, line 2)

Regarding Claim 16, 43, Scheifler discloses the method of claim 1 wherein the called code frame is included within a lower level code assembly following the first code assembly in the runtime call stack. (see Scheifler col. 20, lines 3-7)

Regarding Claims 17- 19, 37- 39, 44- 46, Scheifler discloses the method of claim 1 wherein the operation of associating a first permission grant object with a first code assembly comprises: associating the first permission grant object with an individual method, class, module (method performs functions on a class data structures within a module) of the first code assembly. (see Scheifler col. 11, line 66 - col. 12, line 5)

Regarding Claims 26, 27, Scheifler discloses a runtime system for determining whether a requested permission, requested by a called code frame, is satisfied within a runtime call stack so as to allow the called code frame to perform a protected operation, the runtime system comprising:

a) a first permission grant object associated with a first code assembly in the runtime call stack; (see Scheifler col. 14 , lines 31-37)

b) a second permission grant object associated with a second code assembly in the runtime call stack; (see Scheifler col. 14 , line 62 - col. 15, line 4)

Scheifler does not disclose the generation of an intersection (i.e. common subset) of permissions for a method's code processing software and cache (i.e. save) of the permission set information for later usage without re-generation of permission set information. However, Jerger discloses a runtime system for determining whether a requested permission, requested by a called code frame, is satisfied within a runtime call stack so as to allow the called code frame to perform a protected operation, the runtime system comprising:

c) a cache storing an intersection of permissions provided by the first permission grant object and the second permission grant object, wherein execution of the called code frame is permitted if the requested permission is a subset of the cached permission intersection. (see Jerger col. 35, lines 40-43; col. 58, lines 37-42)

It would have been obvious to one of ordinary skill in the art at the time the invention was made to modify Scheifler to process permission sets by the generation and save of an intersection of permissions for a software code method as taught by Jerger. One of ordinary skill in the art would be motivated to modify Scheifler in order to enhance security in the execution of software using permission sets. (see Jerger col. 2, line 65 - col. 3, line 5)

Conclusion

7. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Kyung H. Shin whose telephone number is (571) 272-3920. The examiner can normally be reached on 9 am - 7 pm.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, David A. Wiley can be reached on (571) 272-3923. The fax phone number for the organization where this application or proceeding is assigned is 703-872-9306.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

KHS
Kyung H Shin
Patent Examiner
Art Unit 2143

KHS
June 25, 2005

DAVID WILEY
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100